A New Approach to Finding 2 x n Partially Spatially Balanced Latin Rectangles

Renee Mirka ✉
Cornell University, Ithaca, NY, USA

Laura Greenstreet ✉
Cornell University, Ithaca, NY, USA

Marc Grimson ✉
Cornell University, Ithaca, NY, USA

Carla P. Gomes ✉
Cornell University, Ithaca, NY, USA

Abstract

Partially spatially balanced Latin rectangles are combinatorial structures that are important for experimental design. However, it is computationally challenging to find even small optimally balanced rectangles, where previous work has not been able to prove optimality for any rectangle with a dimension above size 11. Here we introduce a graph-based encoding for the $2 \times n$ case based on finding the minimum-cost clique of size n. This encoding inspires a new mixed-integer programming (MIP) formulation, which finds exact solutions for the 2×12 and 2×13 cases and provides improved bounds up to $n = 20$. Compared to three other methods, the new formulation establishes the best lower bound in all cases and establishes the best upper bound in five out of seven cases.

2012 ACM Subject Classification Applied computing → Operations research

Keywords and phrases Spatially balanced Latin squares, partially spatially balanced Latin rectangles, minimum edge weight clique, combinatorial optimization, mixed integer programming, imbalance, cliques

Digital Object Identifier 10.4230/LIPIcs.CP.2023.47

Category Short Paper

Funding This material is based upon work partially supported by the National Science Foundation under Awards CCF-1522054 and CCF-2007009 and by AFOSR DURIP grant FA9550-21-1-0316, AFOSR MURI grant FA9550-18-1-0136, and AFOSR grant FA9550-20-1-0421.

1 Introduction

Latin squares and rectangles are combinatorial objects represented by $n \times n$ or $k \times n$ grids with entries assigned from $\{1, 2, \ldots, n\}$ such that no entry is repeated in a row or column. They are important structures for experimental design. For example, in agronomic field experiments, experts are interested in applying n treatments consisting of n fertilizers in different orderings, which can be achieved by designing the treatment sequences following a Latin square. Arbitrary Latin squares are not hard to generate. However, geometric imbalance due to some treatments occurring closer together more frequently can bias experimental results [21]. This motivates the use of *spatially balanced* Latin squares and rectangles which require additional structure to capture the notion of distance between any two treatments in the square or rectangle and are more computationally challenging to construct.

A large body of previous work has focused on spatially balanced Latin squares, including introducing streamlining constraints [8], using stochastic optimization [11, 10, 12, 6], and applying local search methods [22]. While spatially balanced Latin squares have been extensively studied [9, 16, 17, 21], there has been much less work on spatially balanced Latin rectangles [4], despite rectangular conditions occurring more frequently in practice. For example, in the case of fertilizer treatments, the number of treatments is often less than the number of fertilizers resulting in a rectangular structure.

The main previous work investigating spatially balanced Latin rectangles established the non-existence of perfectly balanced Latin rectangles for an infinite family of sizes and shifted focus to constructing *partially* spatially balanced Latin rectangles (PSBLRs) [4]. Diaz et al. introduce and experimentally compare three approaches to generating PSBLRs: an assignment-based mixed-integer program (MIP), a constraint satisfaction program (CP), and a random-restart hill climbing local search. Using these approaches, Diaz et al. were able to find the provably optimal imbalance for rectangles up to a size of 2×11 and provide bounds up to 12×12.

In this work, we focus on the $2 \times n$ case of constructing PSBLRs, introducing a new graph-based encoding based on a reduction to a minimum-edge weight clique problem. The maximum/minimum edge-weight clique problem (MEWC) is a generalization of the classic max clique problem, where given an input graph $G = (V, E)$ instead of finding the largest complete subgraph the goal is to find the subgraph with the largest/smallest sum of weighted edges. The MEWC problem has many applications including in experimental design [3], molecular biology [20], and materials discovery [1]. Multiple approaches have been developed for MEWC problems including linear and quadratic mixed-integer programs [5, 7, 18], branch-and-cut [5, 15, 19], and heuristic methods [2, 14]. Several benchmarks have been developed for MEWC, though less than half the instances have been solved to optimality [13].

For our experiments, we use a straight-forward MIP formulation of MEWC to emphasize the benefits of the reduction instead of advanced techniques developed for MEWC. This new MIP formulation for the problem of finding $2 \times n$ PSBLRs finds optimal solutions for the previously unsolved cases of $n = 12$ and 13 and provides new bounds for $n = 14 - 20$. Further, we demonstrate the new clique-based MIP formulation outperforms the previous assignment-based MIP formulation and A* search and finds comparable solutions to local search while providing lower bounds.

2 Preliminaries

▶ **Definition 1** (Latin Rectangle). *Let k and n be positive integers with $k \leq n$ and R be a $k \times n$ matrix. Then R is a Latin rectangle if every entry of R contains a number in $[n] = \{1, 2, \cdots, n\}$ and no number is repeated in any row or column.*

In order to discuss partially spatially balanced Latin rectangles, we first introduce the notion of distance between two symbols in a Latin rectangle and imbalance:

▶ **Definition 2** (Imbalance). *For two symbols $u, v \in [n]$ and $i \leq k$, the distance between u and v in row i, denoted $d_i(u, v)$, is the absolute value of the difference of the indices of the positions of u and v in row i. The overall distance between u and v is then $d(u, v) = \sum_{i \leq k} d_i(u, v)$.*

The spatial imbalance of a Latin rectangle, R, is defined by

$$\mathbb{I}(R) = \sum_{i,j} \left| d(i,j) - \frac{k(n+1)}{3} \right| .$$

▶ **Definition 3** (Spatially Balanced Latin Rectangle). *We say that a $k \times n$ rectangle R is spatially balanced if all distances $d(u, v)$ are the same. For brevity, we denote this as $SBLR(k, n)$.*

Note that in this case, Diaz et al. show that each distance must be exactly $\frac{k(n+1)}{3}$ and thus the imbalance of the rectangle is 0; see [4] for a short proof of this proposition.

▶ **Proposition 4.** *If there exists a solution for SBLR(k,n) then the distance between any pair of symbols is equal to $k(n+1)/3$.*

As a corollary, $k \equiv_3 0$ or $n \equiv_2 2$ are necessary conditions for the existence of SBLR(k,n) as $k(n+1) \equiv_3 0$ must hold. Diaz et al. have further shown the non-existence of SBLRs of size $2 \times n$ for $n \neq 2$ and $3 \times n$ for $n \neq 3$, as well as experimentally demonstrated that no perfectly spatially balanced rectangles with $k \neq n$ exist up to size 7×7 [4]. In practice, it is useful to minimize the imbalance even if it is not possible to reduce it to zero. This motivates the definition of the object of our study- partially spatially balanced Latin rectangles (PSBLRs):

▶ **Definition 5** (Partially Spatially Balanced Latin Rectangle). *A $k \times n$ Latin rectangle R is partially spatially balanced if $\mathbb{I}(R)$ is minimized. In particular, R is partially spatially balanced if for any $k \times n$ Latin rectangle R', $\mathbb{I}(R) \leq \mathbb{I}(R')$.*

3 Graph Encoding

In this section, we describe a new graph-based encoding for the problem of constructing PSBLRs. For the $2 \times n$ Latin rectangles we consider, we will always assume the first row is ordered and given by $1 \quad 2 \quad \dots \quad n$, as any other solution can be transformed into a solution in this form through relabelling symbols. This reduces the problem to selecting a single imbalance-minimizing derangement, or permutation with no fixed points, for the second row of the rectangle. It also simplifies the computation of the distance between any two symbols u and v. Particularly, for $1 \leq u < v \leq n$, we have $d(u,v) = v - u + |i_v - i_u|$ where i_v, i_u are the indices of v, u in the second row, respectively. For fixed u, v and n, $d(u,v)$ is solely determined by the choices for i_u and i_v. We exploit this through the construction of a graph which encodes how these choices affect the imbalance.

The graph encoding is as follows. For a given n, construct a complete graph $G = (V,E)$ with $V = \{v_{ij} : 1 \leq i,j \leq n, i \neq j\}$ and $E = \{(u,v) : u,v \in V\}$. Intuitively, a vertex v_{ij} represents a Latin rectangle where j is the index of i in the second row. We include a cost on each edge (v_{st}, v_{qr}) which represents how much is contributed to the imbalance from $d(s,q)$ if t and r are the indices of s and q in the second row, respectively. In particular, for $e = (v_{st}, v_{qr}) \in E$ with $s \neq q$ and $t \neq r$, let

$$\begin{aligned} c_e &= \left| d(s,q) - \frac{2(n+1)}{3} \right| \\ &= \left| |s-q| + |i_s - i_q| - \frac{2(n+1)}{3} \right| \\ &= \left| |s-q| + |t-r| - \frac{2(n+1)}{3} \right|. \end{aligned}$$

If $s = q$ or $t = r$, the cost on the edge is ∞, since the assignments these vertices represent are not valid for a Latin rectangle. See Figure 1 for an example when $n = 3$.

To complete the problem encoding, we need to discern how to recover a PSBLR from this complete graph. The key observation is that every n-clique where all edges have finite cost corresponds to a valid Latin rectangle. The finiteness of the costs ensures that the assignment rules of the Latin rectangle are obeyed. Furthermore, the sum of the costs on the edges of an n-clique is exactly equal to the imbalance of its corresponding Latin rectangle. As such, to find a PSBLR, it suffices to find an n-clique whose sum of edge costs is minimal. Figure 1 illustrates an optimal clique corresponding to an optimal Latin rectangle for $n = 3$:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{bmatrix}$$

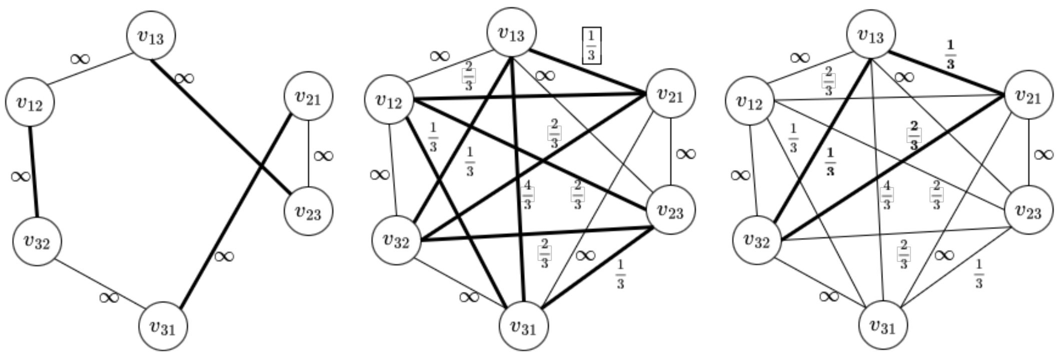

Figure 1 (Left) The graph encoding after including all edges with infinite costs between vertices v_{sq} and v_{sr} for $s = 1, 2, 3$ (thin edges) and between vertices v_{sq} and v_{rq} for $q = 1, 2, 3$ (bold edges). (Center) The graph encoding after including all remaining edges with finite costs (bold edges) between the remaining pairs of vertices. The boxed value is calculated as $|(2-1)+(3-1)-8/3| = 1/3$. (Right) A 3-clique (bold edges) in the graph encoding when $n = 3$ corresponding to a 2×3 PSBLR.

4 MIP Formulation

Now that we have reduced the problem of finding a $2 \times n$ PSBLR to finding a minimum-cost n-clique, we can model the problem through a new integer program:

$$\begin{aligned}
\text{minimize} \sum_{e\in E} c_e x_e & & \\
\text{subject to} \sum_{v\in V} z_v = n, & & (1)\\
x_e \geq z_v + z_u - 1, \quad & \forall e = (u,v) \in E, & (2)\\
x_e \in \{0,1\}, \quad & \forall e \in E, & \\
z_v \in \{0,1\} \quad & \forall v \in V. &
\end{aligned}$$

In this model, we have binary variables z_v and x_e for each vertex v and edge e in the graph representing whether the vertex or edge is included in the n-clique. While (1) guarantees that we select n vertices, (2) requires that any edges between two selected vertices are included in the cost; together these ensure the model selects an n-clique.

Note that the size of this model is polynomial in n with $n(n-1)+n(n-1)(n(n-1)-1)/2$ variables and $n(n-1)(n(n-1)-1)/2+1$ constraints. Furthermore, by construction, an optimal solution will be a PSBLR. However, the presence of so many binary variables makes finding the optimal solution a cumbersome computational task.

The following observation allows us to partially relax the integer program and significantly reduce the number of binary variables:

▶ **Observation 6.** *For each edge $e \in E$, x_e will be 0 or 1 if z_v is binary for all $v \in V$. In other words, we can relax the restriction that x_e is binary for all $e \in E$ by replacing it with $x_e \geq 0$ for all $e \in E$ and still guarantee a binary solution.*

The relaxed mixed integer program only requires $n(n-1)$ binary variables, one for each vertex, where the remaining edge variables are continuous.

5 Methods

We compared the min-cost n-clique MIP formulation, hereafter referred to as the clique formulation, to an assignment-based MIP formulation, A* search, and a local search approach.

In the assignment based MIP formulation, we again assume the first row is fixed to $1 \quad 2 \quad \dots \quad n$. Thus, we only need to introduce binary variables indicating whether value i occurs in position j of the second row, $i, j \in \{1, 2, \cdots, n\} \equiv [n], i \neq j$. Note that these exactly correspond to the vertex variables in the clique formulation, $z_v = z_{v_{ij}}$ for $v \in \{v_{ij} : i, j \in [n], i \neq j\}$. As the imbalance is dependent on pairwise positions in the second row, we also introduce binary variables that are 1 if a pair of variables is included in the solution. Note that these correspond exactly to the edge variables in the clique formulation, x_e for $e \in \{(u, v) : u, v \in V\}$, where the pairwise cost corresponds to c_e. Finally, we introduce constraints ensuring each value appears exactly once in the second row (3) and each position in the second row is assigned exactly one value (4):

$$
\begin{aligned}
\text{minimize} \sum_{e\in E} c_e x_e & & \\
\text{subject to} \sum_{j\in[n], j\neq i} z_{v_{ij}} &= 1, & \forall i \in [n] \qquad (3)\\
\sum_{i\in[n], i\neq j} z_{v_{ij}} &= 1, & \forall j \in [n] \qquad (4)\\
x_e &\geq z_v + z_u - 1, & \forall e = (u, v) \in E,\\
x_e &\in \{0, 1\}, & \forall e \in E,\\
z_v &\in \{0, 1\} & \forall v \in V.
\end{aligned}
$$

Similar to the clique formulation, we can partially relax the integer program by allowing the x_e variables to assume continuous values, reducing the number of binary variables to $n(n-1)$.

Both MIP formulations were implemented in Gurobi and CPLEX using their respective Python APIs. As previous work has established optimal solutions up to $n = 11$, we replicated previous results and further tested instances with $n = 12 - 20$. For each n, both models were tested for 6 hours running on a Intel Xeon 6154 processor and allowed 32GB of memory in three configurations: a single instance running on a single thread, a single instance running on 32 threads, and 8 instances running on a total of 32 threads.

For A* search, we again fix the first row and only consider positions in the second row. Starting with an empty second row, we add all valid placements of the first digit to a priority queue, assigning each node a cost based on the partial imbalance of the filled digits and an admissible heuristic for the imbalance due to the unplaced digits. We repeatedly evaluate the first node in the queue until we reach a node with all digits filled which is guaranteed to be an optimal solution. As a heuristic, for each unplaced digit we calculated the minimum increase to the imbalance that would result from placing it in one of the remaining positions, accounting only for interactions with already placed digits. We then summed these values across all unplaced digits.

A* search was implemented in C++ and was also tested for 6 hours each on instances of size $n = 12 - 20$ on an Intel Xeon 6154 processor. As A* search is memory intensive, runs were allocated 128GB of memory, compared to 32GB for the other methods. If A* search does not complete, the value of the node at the front of the queue can be used as a lower bound on the solution value. However if the algorithm does not run to completion, no feasible solutions are found, resulting in no upper bound on the imbalance.

We implemented local search using a random walk from a random initial derangement, where at each step the position of two random digits were swapped, maintaining feasibility. The random walk was implemented in C++ and tested for 6 hours each on instances of size $n = 12 - 20$ on an Intel Xeon 6154 processor and allocated 32GB of memory. Unlike the other methods, a random walk cannot prove optimality or provide a lower bound on the solution.

In addition to testing methods to bound the optimal solution, we performed several tests to better characterize the solution space. First, for $n = 2 - 10$ we brute force computed all solutions to determine the number of optimal solutions. We were inspired to characterize the number of solutions due to the observation that mirroring either the first or second row results in a rectangle with the same imbalance. For example if we have a solution R, we can construct a rectangle R' with the second row mirrored by setting $R'_{2i} = n + 1 - R_{2i}$ for $i \in [n]$. This results in the same imbalance:

$$\begin{aligned}
\mathbb{I}(R') &= \sum_{i,j} \left| |R_{1i} - R_{1j}| - |(n+1-R_{2i}) - (n+1-R_{2j})| - \frac{2(n+1)}{3} \right| \\
&= \sum_{i,j} \left| |R_{1i} - R_{1j}| - |R_{2i} - R_{2j}| - \frac{2(n+1)}{3} \right| \\
&= \mathbb{I}(R)
\end{aligned}$$

However, it is not guaranteed that R' is a Latin rectangle, as the rearrangement may not respect the column constraint. Similarly, we can create a rectangle, $R^{\dagger}$, with the first row mirrored by setting $R^{\dagger}_{1i} = n + 1 - R_{1i}$ for $i \in [n]$. While this does not result in a rectangle with the first row ordered as $1 \quad 2 \quad \dots \quad n$, we can relabel the variables so the first row goes from $1 \quad 2 \quad \dots \quad n$ which results in the second row having the labels $R^{\dagger}_{2i} = R_{2(n+1-i)}$. While this could potentially allow up to four solutions from a single solution, there is no guarantee that these solutions satisfy the column constraints. As multiple solutions are not guaranteed, we cannot use methods like streamlining. However, the presence of multiple solutions may make it difficult for the MIP formulations to prove optimality.

Further, we sought to characterize the distribution of feasible solutions. For $n = 2 - 10$ we were able to compute the distribution exactly. For $n = 11 - 16$, we randomly sampled a million solutions to approximate the distribution. The distribution of solutions impacts all four methods, where having many solutions with nearly optimal imbalance will benefit a random walk, while it will make it difficult for the MIP formulation and A* search to prove optimality.

6 Results

While CPLEX had better runtimes up to $n = 10$, only Gurobi was able to prove optimality for $n = 12$ and 13 and Gurobi consistently found better bounds for $n = 14 - 20$ (see Tables 3-5). Thus, for the remainder of the results we report the performance of the Guorbi model for both MIP formulations. In general, running the model using a single instance and a single thread performed better than running a single instance with multiple threads or multiple instances, with the only notable exception being for $n = 20$, where running with multiple instances found the smallest upper bound of 816 (See Tables 6-7).

While both MIP formulations were able to prove optimality for $n = 12$, only the clique formulation was able to prove optimality for $n = 13$ within the 6 hours allotted (See Table 8). While local search is not able to prove optimality, it found the optimal solution for both

$n = 12$ and 13. For all $n = 2 - 20$, the clique formulation resulted in the largest lower bound (See Table 1). The results were more varied for the upper bound, where the random walk found the lowest upper bound for $n = 14$ and 17, the clique found the lowest upper bound for $n = 18, 19$, and 20, and both methods found the same upper bound for $n = 15$ and 16 (See Table 2). For $n = 16$, the assignment formulation matched the upper bound found by the clique formulation and random walk.

A* was only able to run to completion for $n = 12$. For $n = 13 - 15$, A* ran out memory, despite being allotted 128GB where all other methods used under 32GB. For $n > 15$, A* was only able to reach configurations with at most seven fixed values, resulting in a weak lower bound on the optimal solution.

Table 1 Lower bound by method for $n = 12 - 20$. Bold values indicate the best bound, i.e. the largest lower bound, for each n. All methods were allotted 6 hours.

	n								
	12	13	14	15	16	17	18	19	20
Clique MIP	**168**	$\mathbf{218.\overline{66}}$	**191**	$\mathbf{252.\overline{66}}$	$\mathbf{273.\overline{66}}$	**252**	$\mathbf{402.\overline{66}}$	$\mathbf{465.\overline{33}}$	**155**
Assignment MIP	**168**	$\mathbf{218.\overline{66}}$	131	$141.\overline{33}$	133	91	121	$138.\overline{66}$	47
A* search	**168**	191	166	169	$164.\overline{66}$	134	159.33	$158.\overline{66}$	106
Random walk	-	-	-	-	-	-	-	-	-

Table 2 Upper bound by method for $n = 12 - 20$. Bold values indicate the best bound, i.e. the smallest upper bound, for each n. All methods were allotted of 6 hours.

	n								
	12	13	14	15	16	17	18	19	20
Clique MIP	**168**	$\mathbf{218.\overline{66}}$	272	$\mathbf{345.\overline{33}}$	$\mathbf{427.\overline{33}}$	522	$\mathbf{617.\overline{33}}$	**732**	**816**
Assignment MIP	**168**	$\mathbf{218.\overline{66}}$	272	346	$\mathbf{427.\overline{33}}$	526	$621.\overline{33}$	**732**	886
A* search	**168**	-	-	-	-	-	-	-	-
Random walk	**168**	$\mathbf{218.\overline{66}}$	**268**	$\mathbf{345.\overline{33}}$	$\mathbf{427.\overline{33}}$	**508**	$623.\overline{33}$	$742.\overline{66}$	866

For the exploratory tests, for all $n = 2 - 10$ there are multiple optimal solutions (See Figure 2). The number of solutions did not show any obvious patterns, such as a monotonic increase or alternation between odd and even digits. For $n = 6 - 10$ where we were able to compute the distribution of solutions exactly, up to 10% of solutions were within 10% of optimal and as many as 40% of solutions were within 20% of optimal (See Figure 2). For $n = 11 - 20$ we looked at the distribution over a million random solutions instead of the full solution space, so the best solution likely does not represent the optimal solution. However, the trend of a large proportion of the cumulative distribution being within 20% of the best solution continued, where at the high-end of the range over 40% of solutions were within 10% of the best solution and over 90% were within 20% of the best solution.

7 Discussion

Both the clique and assignment formulation were able to prove optimality for the previously unsolved case of $n = 12$ and the clique formulation was further able to prove optimality for $n = 13$. For $n = 14 - 20$, the clique formulation consistently found the best lower-bounds, outperforming the assignment formulation and A* in all seven cases. While the clique formulation and a random walk were comparable in establishing upper bounds, finding the

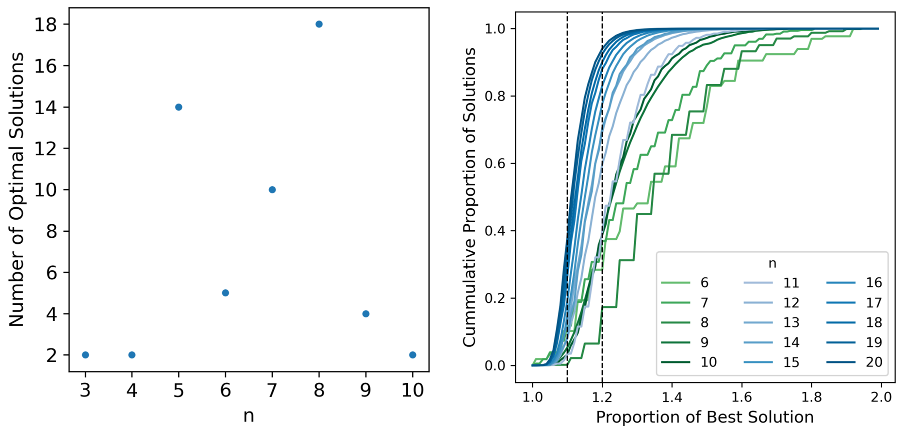

Figure 2 (Left) Number of optimal solutions by n. (Right) Cumulative distribution of solutions as proportion of the best solution. For $n = 6 - 10$ (green), distributions represent all solutions. For $n = 11 - 20$ (blue), distributions are over a million random solutions. Dashed lines indicate solutions that are 10% and 20% greater than the best solution respectively.

best upper bound in five and four cases respectively, the clique formulation appeared to perform better for large n, establishing the best upper bound for $n = 18 - 20$. Thus, the clique formulation outperformed each of the other three methods individually, as well as performed better than using a combination of other methods, such as using A* to establish lower bounds and a random walk to establish upper bounds.

While the clique formulation outperformed the other methods, no method was able to prove the optimality of a solution above $n = 13$. The exploratory results shed some light on why it is difficult to find even relatively small PSBLRs. First, the potential for mirror solutions as well as the presence of multiple solutions for all $n = 3 - 10$ make it challenging for the MIP formulations to prove optimality, as potentially many branches of the branch-and-bound tree include optimal solutions and must be extensively explored before they can be pruned. Further, while there are likely multiple solutions, the possibility of a mirror rectangle violating a column constraint means we cannot guarantee multiple solutions and use techniques like streamlining constraints to guide the model towards a single optimal solution.

The large portion of the cumulative distribution close to the optimal solution further hampers the MIP formulations, making it difficult to prune branches that include near-optimal solutions. This large number of near-optimal solutions also hampers A* search, where a very tight heuristic is needed to allow for any significant amount of pruning, where experimentally we found that most nodes could generally only be pruned once all but one or two values had been assigned. While a random walk would benefit from multiple optimal solutions, a random walk or other local-search based methods cannot establish optimality.

8 Conclusion

Spatially and partially spatially balanced Latin squares and rectangles are important combinatorial structures used for experimental design. While spatially balanced Latin squares have been extensively studied, relatively little work has considered spatially balanced Latin rectangles, despite them occurring more frequently in practice.

We introduce a new graph-based encoding for a $2 \times n$ Latin rectangle which inspires a new MIP formulation based on the MEWC problem. This new formulation found optimal solutions previously unsolved 2×12 and 2×13 cases and outperforms an assignment-based MIP formulation, A* search, and a random-walk based local-search, establishing improved bounds up to $n = 20$. Given the success of our straight-forward MIP implementation for MEWC, a direction for future work is to explore whether more advanced MEWC methods are able to find optimal PSBLRs for $n > 13$.

Further, our exploratory results help characterize what make finding PSBLRs computationally challenging. The potential, but not guarantee, of multiple optimal solutions makes it difficult for mathematical programming methods to establish optimality and prevents the use of standard methods for handling multiple solutions, such as streamlining. The large number of near-optimal solutions makes it difficult for informed search-based methods to prune any significant amount of solutions, requiring near brute-force exploration of the search space. While simple to encode, this problem is quite challenging, making it an ideal benchmark for future search and optimization methods and we encourage further exploration of the problem.

References

1 Luis A Agapito, Marco Fornari, Davide Ceresoli, Andrea Ferretti, Stefano Curtarolo, and Marco Buongiorno Nardelli. Accurate tight-binding hamiltonians for two-dimensional and layered materials. *Physical Review B*, 93(12):125137, 2016.

2 Bahram Alidaee, Fred Glover, Gary Kochenberger, and Haibo Wang. Solving the maximum edge weight clique problem via unconstrained quadratic programming. *European Journal of Operational Research*, 181(2):592–597, 2007.

3 Egon Balas and Chang Sung Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4):1054–1068, 1986.

4 Mateo Díaz, Ronan Le Bras, and Carla Gomes. In search of balance: The challenge of generating balanced latin rectangles. In *Proceedings of Integration of AI and OR Techniques in Constraint Programming: 14th International Conference*, pages 68–76. Springer, 2017.

5 G Dijkhuizen and Ulrich Faigle. A cutting-plane approach to the edge-weighted maximal clique problem. *European Journal of Operational Research*, 69(1):121–130, 1993.

6 Stefano Ermon, Carla Gomes, Ashish Sabharwal, and Bart Selman. Low-density parity constraints for hashing-based discrete integration. In *Proceedings of International Conference on Machine Learning*, pages 271–279. PMLR, 2014.

7 U Faigle, R Garbe, K Heerink, and B Spieker. Lp—relaxations for the edge—weighted subclique problem. In *Operations Research'93: Extended Abstracts of the 18th Symposium on Operations Research held at the University of Cologne September 1–3, 1993*, pages 157–160. Springer, 1994.

8 Carla Gomes and Meinolf Sellmann. Streamlined constraint reasoning. In *Proceedings of Principles and Practice of Constraint Programming: 10th International Conference*, pages 274–289. Springer, 2004.

9 Carla Gomes, Meinolf Sellmann, Cindy Van Es, and Harold Van Es. The challenge of generating spatially balanced scientific experiment designs. In *Proceedings of the International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 387–394. Springer, 2004.

10 Carla P Gomes, Joerg Hoffmann, Ashish Sabharwal, and Bart Selman. Short xors for model counting: from theory to practice. In *Proceedings of Theory and Applications of Satisfiability Testing–SAT 2007: 10th International Conference*, pages 100–106. Springer, 2007.

11 Carla P Gomes, Ashish Sabharwal, and Bart Selman. Model counting: A new strategy for obtaining good bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 10, pages 1597538–1597548, 2006.

12 Carla P Gomes, Willem Jan van Hoeve, Ashish Sabharwal, and Bart Selman. Counting csp solutions using generalized xor constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 204–209, 2007.

13 Seyedmohammadhossein Hosseinian, Dalila BMM Fontes, Sergiy Butenko, Marco Buongiorno Nardelli, Marco Fornari, and Stefano Curtarolo. The maximum edge weight clique problem: formulations and solution approaches. *Optimization Methods and Applications: In Honor of Ivan V. Sergienko's 80th Birthday*, pages 217–237, 2017.

14 Seyedmohammadhossein Hosseinian, DBMM Fontes, and Sergiy Butenko. A quadratic approach to the maximum edge weight clique problem. In *XIII Global Optimization Workshop GOW*, volume 16, pages 125–128, 2016.

15 Marcel Hunting, Ulrich Faigle, and Walter Kern. A lagrangian relaxation approach to the edge-weighted clique problem. *European Journal of Operational Research*, 131(1):119–131, 2001.

16 Ronan Le Bras, Carla Gomes, and Bart Selman. From streamlined combinatorial search to efficient constructive procedures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26-1, pages 499–506, 2012.

17 Ronan Le Bras, Andrew Perrault, and Carla P Gomes. Polynomial time construction for spatially balanced latin squares. Technical report, Cornell University, 2012.

18 Anuj Mehrotra. Cardinality constrained boolean quadratic polytope. *Discrete Applied Mathematics*, 79(1-3):137–154, 1997.

19 Michael M Sørensen. New facets and a branch-and-cut algorithm for the weighted clique problem. *European Journal of Operational Research*, 154(1):57–70, 2004.

20 Etsuji Tomita, Tatsuya Akutsu, and Tsutomu Matsunaga. *Efficient algorithms for finding maximum and maximal cliques: Effective tools for bioinformatics*. IntechOpen, 2011.

21 HM Van Es and CL Van Es. Spatial nature of randomization and its effect on the outcome of field experiments. *Agronomy Journal*, 85(2):420–428, 1993.

22 Pascal Van Hentenryck and Laurent Michel. Differentiable invariants. In *Proceedings of Principles and Practice of Constraint Programming: 12th International Conference*, pages 604–619. Springer, 2006.

A Supplementary Results

Table 3 Runtime in seconds for single instances of the clique formulation run in Gurobi and CPLEX using a single thread for $n = 8 - 13$. The dash indicates that the model was not able to prove optimality in 6 hours (21,600 s). For all n less than 8, both models ran in under one second.

	n					
	8	9	10	11	12	13
CPLEX	**1.10**	**4.48**	**28.76**	675.16	-	-
Gurobi	1.15	4.99	51.13	**171.22**	**614.22**	**4750.96**

	n						
	14	15	16	17	18	19	20
CPLEX	132	119.29	97.54	88	47.79	40.0$\overline{6}$	45.06
Gurobi	**191**	**252.66**	**273.$\overline{66}$**	**252**	**402.$\overline{66}$**	**465.$\overline{33}$**	**155**

Table 4 Comparison of the greatest lower bound found by single instances of the clique formulation run in Gurobi and CPLEX using a single thread for 6 hours for $n = 14 - 20$.

	n						
	14	15	16	17	18	19	20
CPLEX	**272**	354	437.$\overline{33}$	526	631.$\overline{33}$	762	878
Gurobi	**272**	**345.$\overline{33}$**	**427.$\overline{33}$**	**522**	**617.$\overline{33}$**	**732**	**876**

Table 5 Comparison of the smallest upper bound found by single instances of the clique formulation run in Gurobi and CPLEX using a single thread for 6 hours for $n = 14 - 20$.

		n						
Instances	Threads/Inst.	14	15	16	17	18	19	20
1	1	191	**252.$\overline{66}$**	273.$\overline{66}$	**252**	**402.$\overline{66}$**	**465.$\overline{33}$**	**155**
8	1	159.69	185.28	203.11	198.56	-	389.50	92.5
1	32	**201.57**	249.79	**273.82**	240.90	185.75	371.95	28

Table 6 Comparison of the greatest lower bound found by three Gurobi configurations run for 6 hours for $n = 14 - 20$. The dashed cell indicates a run that failed due to numerical instability.

		n						
Instances	Threads/Inst.	14	15	16	17	18	19	20
1	1	272	**345.$\overline{33}$**	**427.$\overline{33}$**	**522**	**617.$\overline{33}$**	**732**	886
8	1	**270**	358.$\overline{66}$	442.$\overline{66}$	530	-	747.$\overline{33}$	**816**
1	32	276	347.$\overline{33}$	430.$\overline{66}$	**522**	631.$\overline{33}$	766	900

Table 7 Comparison of the smallest upper bound found by three Gurobi configurations run for 6 hours for $n = 14 - 20$. The dashed cell indicates a run that failed due to numerical instability.

	n					
	8	9	10	11	12	13
Assignment MIP	1.09	6.23	59.02	678.39	3790.28	-
Clique MIP	1.15	4.99	51.13	171.22	**614.22**	**8573.43**
A*	**0.07**	**0.78**	**8.72**	**98.70**	1299.66	-

Table 8 Runtime in seconds for single instances of the clique and assignment formulations run in Gurobi using a single thread and A* search for $n = 8 - 13$. The dash indicates that the model was not able to prove optimality in 6 hours (21,600 s). For all n less than 8, all models finished in under one second. The random walk is not included in this table as the method does not prove optimality.